

Graph based Version of KNN and AHC Algorithm in Combining Text Summarization with Text Clustering

Taeho Jo
President
Alpha AI Research
Cheongju, South Korea
tjo018@naver.com

Abstract—In this research, we propose the graph based KNN algorithm and the AHC algorithm for improving the text summarization automation, using the text clustering. In the previous work, even if the KNN version was applied to the text summarization, a domain should be determined as an input text tag in advance, manually. In this research, text subgroups are built through the text clustering, and each paragraph as an input text partition is classified by selecting a classifier which corresponds to its most similar cluster. The AHC algorithm and the KNN algorithm which processes graphs are adopted respectively as the approaches to the text clustering and the text summarization. The goals of this research are to automate the text summarization and to improve the performance of both tasks, using the graph-based algorithms.

I. INTRODUCTION

The task which is covered in this research is the text summarization where the essential part is extracted from a text as its summary. The text summarization is interpreted into the binary classification of each paragraph into summary or non-summary, to judge whether each partition is essential or not. The process of text summarization is to partition an input text into paragraphs, classify each paragraph into one of the two categories, and extract paragraphs which are classified into summary. The text summarization is viewed as several independent binary classification tasks, each of which corresponds to the domain; an independent classifier is allocated to each domain. It is necessary to present the domain for nominating a most suitable classifier in the text summarization system.

This research is motivated by the necessity of defining domains manually for designing the text summarization system. The text summarization is mapped into a binary classification of paragraphs domain by domain; it is decomposed with binary classification tasks as many as domains. The process of designing the text summarization system is to predefine domains depending on prior knowledge and subjective views, collect sample paragraphs domain by domain, and allocate a binary classifier for each domain. One more motivation of this research is the validation of the graph based KNN algorithm and the graph based AHC algorithm as the excellent approaches to the text summarization and the text clustering. This research is intended to automate the

domain predefinition by connecting the text summarization with the text clustering.

The idea of this research is to apply the graph based KNN algorithm and the graph based AHC algorithm to the text summarization and the text clustering which relate to each other. In this research, we prepare the text collection where a summary is assigned to each text. The similarity metric between two graphs is defined, and the KNN algorithm and the AHC algorithm are modified into the graph-based versions as the approaches to both tasks. The role of text clustering is to segment the text collection into subgroups each of which consists of content based similar texts, and in each subgroup, which indicates a domain, sample paragraphs which are labeled with summary or non-summary are collected, and a binary classifier is trained with them. A domain is automatically decided to an input text by its similarities with the cluster prototypes.

Let us mention some benefits from this research. The main problems in encoding texts into numerical vectors by encoding them into graphs. The better performances in the text summarization and the text clustering are expected by adopting the graph-based versions by the KNN algorithm and the AHC algorithm. We do not need to define the domains depending on prior knowledge and subjective views by automating the domain predefinition through the text clustering. We do not need to decide manually a domain of an input text by computing its similarities with the cluster prototypes.

Let us mention some remaining tasks as further discussions on this research. More advanced machine learning algorithms and deep learning algorithms are modified into graph-based versions. We need to define and characterize mathematically other operations on graphs. We implement the text summarization system which relate to the text clustering as a real program or a library. The text classification and the text clustering may relate to the text summarization for improving their performances.

II. PREVIOUS WORKS

Let us survey the previous cases of encoding texts into structured forms for using the machine learning algorithms

to text mining tasks. The three main problems, huge dimensionality, sparse distribution, and poor transparency, have existed inherently in encoding them into numerical vectors. In previous works, various schemes of preprocessing texts have been proposed, in order to solve the problems. In this survey, we focus on the process of encoding texts into alternative structured forms to numerical vectors. In other words, this section is intended to explore previous works on solutions to the problems.

Let us mention the popularity of encoding texts into numerical vectors, and the proposal and the application of string kernels as the solution to the above problems. In 2002, Sebastiani presented the numerical vectors are the standard representations of texts in applying the machine learning algorithms to the text classifications [1]. In 2002, Lodhi et al. proposed the string kernel as a kernel function of raw texts in using the SVM (Support Vector Machine) to the text classification [2]. In 2004, Lesile et al. used the version of SVM which proposed by Lodhi et al. to the protein classification [3]. In 2004, Kate and Mooney used also the SVM version for classifying sentences by their meanings [4].

It was proposed that texts are encoded into tables instead of numerical vectors, as the solutions to the above problems. In 2008, Jo and Cho proposed the table matching algorithm as the approach to text classification [5]. In 2008, Jo applied also his proposed approach to the text clustering, as well as the text categorization [7]. In 2011, Jo described as the technique of automatic text classification in his patent document [6]. In 2015, Jo improved the table matching algorithm into its more stable version [11].

Previously, it was proposed that texts should be encoded into string vectors as other structured forms. In 2008, Jo modified the k means algorithm into the version which processes string vectors as the approach to the text clustering[7]. In 2010, Jo modified the two supervised learning algorithms, the KNN and the SVM, into the version as the improved approaches to the text classification [8]. In 2010, Jo proposed the unsupervised neural networks, called Neural Text Self Organizer, which receives the string vector as its input data [9]. In 2010, Jo applied the supervised neural networks, called Neural Text Categorizer, which gets a string vector as its input, as the approach to the text classification [10].

The above previous works proposed the string kernel as the kernel function of raw texts in the SVM, and tables and string vectors as representations of texts, in order to solve the problems. Because the string kernel takes very much computation time for computing their values, it was used for processing short strings or sentences rather than texts. In the previous works on encoding texts into tables, only table matching algorithm was proposed; there is no attempt to modify the machine algorithms into their table based version. In the previous works on encoding texts into string vectors, only frequency was considered for defining features

of string vectors. Words which are used as features of numerical vectors which represent texts have their semantic similarities among them, so the similarities will be used for processing sparse numerical vectors, in this research.

III. PROPOSED WORKS

A. Encoding Process

This section is concerned with the graph as a text representation and the similarity between graphs. In representing a text into a graph, a word is given as a vertex, and a similarity between words is given as an edge. A graph is viewed as a set of edges, and the similarity between edges is computed before computing the similarity between graphs. The similarity between them is computed by averaging the similarities of all possible edge pairs. This section is intended to describe the process of encoding a text into a graph and the process of computing the similarity between graphs.

The process of encoding a text into a graph is illustrated in Figure 1. Words are retrieved as the vertices from the text by indexing it. The similarities among words are computed as edges in the edge definition. In the edge selection, the edges with their higher similarities are selected for representing a text into a graph. Refer to [12] for studying the encoding process in more detail.

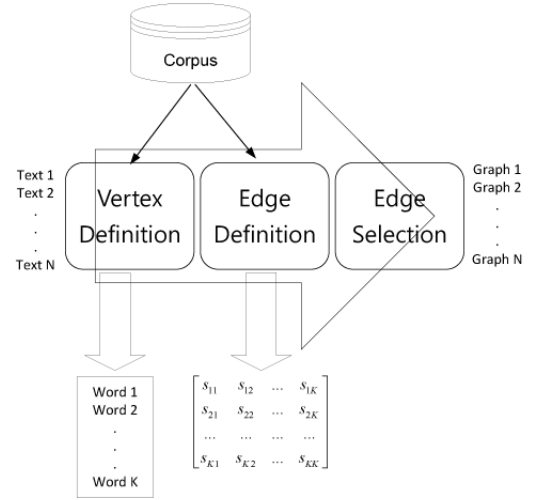


Figure 1. Process of Encoding Texts into Graphs

The three cases of computing the similarity between two edges are illustrated in Figure 2. Each weight which is associated with its own edge is assumed to be a normalized value between zero and one, and if both nodes are identical to each other, the average over two weights is the similarity between two edges. If either of the two nodes is identical to each other, the product of two weights is the similarity between two edges. If neither of two nodes is identical, zero is the similarity between them. The similarity between two edges is always a normalized value between zero and one.

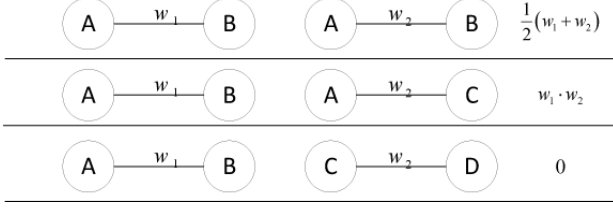


Figure 2. Three Cases of Comparing Two Edges with Each Other

Let us mention the process of computing the similarity between graphs as a semantic similarity between texts. The two graphs which represent texts are notated by edge sets, $G_1 = \{e_{11}, e_{12}, \dots, e_{1|G_1|}\}$ and $G_2 = \{e_{21}, e_{22}, \dots, e_{2|G_2|}\}$. Two edges, $e_{1i} \in G_1$ and $e_{2j} \in G_2$, are associated with their own weights, w_{1i} and w_{2j} , and the similarity between two edges, e_{1i} and e_{2j} by equation (1), (2), or (3), depending on the cases which are mentioned above,

$$\text{sim}(e_{1i}, e_{2j}) = \frac{1}{2}(w_{1i} + w_{2j}) \quad (1)$$

$$\text{sim}(e_{1i}, e_{2j}) = w_{1i} \times w_{2j} \quad (2)$$

$$\text{sim}(e_{1i}, e_{2j}) = 0 \quad (3)$$

The similarity between two graphs is computed by equation (4),

$$\text{sim}(G_1, G_2) = \frac{1}{|G_1| \times |G_2|} \sum_{i=1}^{|G_1|} \sum_{j=1}^{|G_2|} \text{sim}(e_{1i}, e_{2j}). \quad (4)$$

The value which is generated from equation (4), is always given as a normalized value between zero and one as shown in equation (5),

$$0 \leq \text{sim}(G_1, G_2) \leq 1. \quad (5)$$

Let us make some remarks on the encoding process and the computation of the similarity between texts. A text is encoded into a graph by the vertex definition, the edge definition, and the edge weighting. The three cases are considered for computing the similarity between edges. The similarity between graphs is computed by equation (4). In the future research, we consider representing a text into multiple graphs.

B. Graph based KNN Algorithm

This section is concerned with the graph based KNN algorithm. It was initially proposed as the approach to the word classification by Jo in 2018 ???. The similarity metric between graphs which was described in the previous section is used for computing the similarity between a novice graph and a training example. The labels of its nearest neighbors are voted with their identical weights for deciding the label of a novice input. This section is intended to describe the initial version of KNN algorithm from which its variants are derived.

The process of computing the similarity between a novice item and a training example is illustrated in Figure 3. The labeled words which are gathered as samples are encoded into graphs, and they are notated by a set, $Tr = \{G_1, G_2, \dots, G_N\}$. A novice word is encoded into a graph notated by G . Its similarities with the graphs which represent the sample words are computed by equation (4), as $\text{sim}(G, G_1), \text{sim}(G, G_2), \dots, \text{sim}(G, G_N)$. Some training examples which are most similar as the novice input are selected as its nearest neighbors.

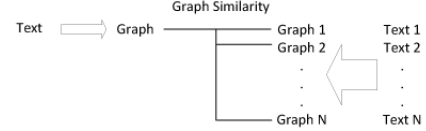


Figure 3. Similarities of Novice Input with Training Examples

In Figure 4, the process of selecting nearest neighbors by the ranking selection is illustrated. The training examples are sorted by the descending order of their similarities, after computing their similarities with a novice example. The training examples with their higher similarities with a novice example are selected as its nearest neighbors, as expressed in equation (6),

$$Nr = \text{Select}_k(Tr, G), Nr \subseteq Tr \quad (6)$$

The set of nearest neighbors, Nr , is composed with elements as expressed in equation (7),

$$Nr = \{G_1^{near}, G_2^{near}, \dots, G_k^{near}\} \quad (7)$$

The elements in the set, Nr , are used for voting their labels in the next step.

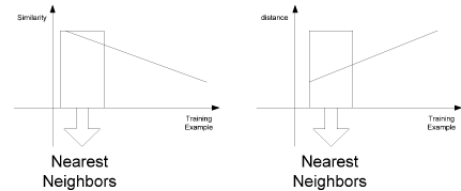


Figure 4. Selection of Most Similar or Least Distant Training Examples as Nearest Neighbors

The process of voting the labels of nearest neighbors for decoding the label of a novice word is illustrated in Figure 5. The predefined categories are notated by $c_1, c_2, \dots, c_m$, and the label of a nearest neighbor is notated by $c_j = \text{label}(G_i^{near})$. The number of nearest neighbors which belong to the category, c_j , is notated by $\text{Count}(Nr, c_j)$. The label of the novice item, G , is decided by equation (8),

$$\text{label}(G) = \arg\max_{j=1}^m \text{Count}(Nr, c_j) \quad (8)$$

The process of deciding the label of a novice item by equation (8), is called voting.

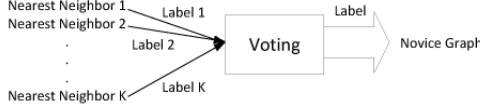


Figure 5. Voting of Labels of Nearest Neighbors for Deciding Label of Novice Input

Let us make some remarks on the graph based KNN algorithm. It computes the similarities of a novice item with the training examples. The training examples with their highest similarities with the novice item are selected as its nearest neighbors. The label of the novice item is decided by voting the labels of its nearest neighbors. The ranking selection is adopted for selecting the nearest neighbors from training examples, in this version.

C. System Architecture

This section is concerned with the text summarization system which adopts the graph based KNN algorithm. In Section III-B, we described the proposed version of KNN algorithm as the approach to the text summarization. It is viewed as a binary classification which classifies a paragraph into summary or non-summary. This section is intended to describe the text summarization system with respect to its function and its architecture.

The process of sampling paragraphs domain by domain is illustrated in Figure 6. Even if it is possible map the text summarization into an instance of text classification, a paragraph may be classified differently depending on the domain. Sample paragraphs which are labeled with summary or non-summary are gathered domain by domain. The text which is given as the input is tagged with a domain, and the classifier which corresponds to the domain is appointed. The sample paragraphs are encoded into graphs in each domain.

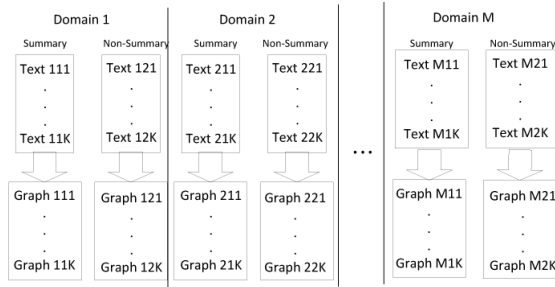


Figure 6. Preparation of Training Examples

The entire architecture of the proposed text summarization system is illustrated in Figure 7. A text is given as the input, and paragraphs are extracted by partitioning the text. In the encoding module, the sample paragraphs in the summary group and the non-summary group and ones which are extracted from the text are mapped into graphs in the encoding module. The paragraphs from the text are classified

into one of the two categories in the similarity computation module and the voting module. The paragraphs which are classified into summary are extracted as the output in this system.

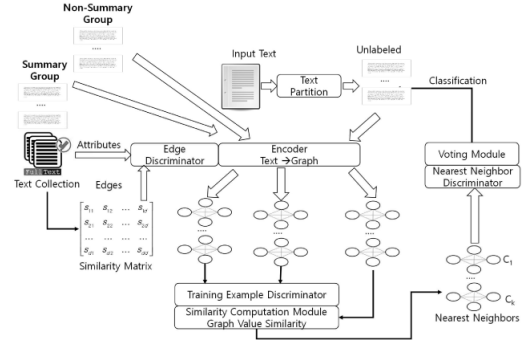


Figure 7. System Architecture

The execution flow of the text summarization system is illustrated in Figure 8. In each domain, sample paragraphs which are labeled with summary or non-summary are gathered. The input text is partitioned into novice paragraphs, and both sample paragraphs and novice paragraphs are encoded into graphs. Each paragraph is classified into summary or non-summary, and there are two groups of paragraphs in the text: summary group and non-summary group. The paragraph which are classified into summary are extracted as the summary of the input text in this system.

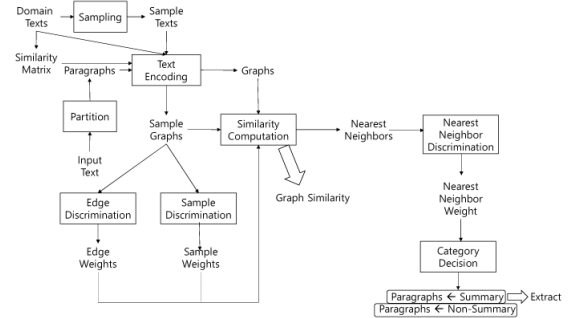


Figure 8. System Flow

Let us make some remarks on the proposed system which is illustrated in Figure 7 as its architecture. The text summarization is defined as the binary classification where each paragraph is classified into summary or non-summary. Paragraphs are encoded into graphs instead of numerical vectors, and they are classified directly. Ones which are classified with summary are selected as the summary of the input text. We need to add the text classification module for predicting the domain of input text automatically.

D. Connection with Text Clustering

This section is concerned with the connection of the text summarization with the text clustering. In the previous

section, we mentioned the text summarization as an instance of domain dependent classification. The domains are automatically constructed from the text collection by clustering texts. The AHC algorithm which is proposed in [13] is used for implementing the text clustering. This section is intended to describe the text summarization system which is connected with the text clustering for automating the construction of domains.

The architecture of the text clustering system which was proposed in [13] is illustrated in Figure 9. The texts in the group are encoded into graphs, and the AHC algorithm which is proposed in [13] is adopted as the approach. It starts with singletons as many as items, computes the similarities of all possible cluster pairs, and merge the cluster pair with its highest similarity into one cluster. The two steps are iterated until the number of clusters reach the desired number. We may consider replacing the AHC algorithm by its variants for improving the speed and the performance.

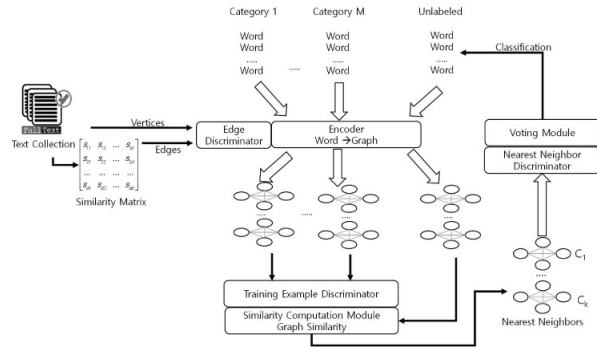


Figure 9. Text Clustering System

The connection of the text summarization with the text clustering is illustrated in Figure 10. The M domains are defined by clustering texts in a group, initially and automatically. A novice text is given as the input, and its domain, domain D , is decided by its similarities with domains. A classifier which corresponds to domain D is nominated and classify each paragraph in a novice text into summary or non-summary. The M text summarization systems are viewed as independent ones, each of which classifies each paragraph into one of the two categories.

The execution flow of text summarization which is connected with the text clustering is illustrated in Figure 11. Unlabeled texts are clustered into subgroups, each of which is a domain. Sample paragraphs which are labeled with summary or non-summary are generated from each domain. These sample paragraphs are provided for training the classifier in each text summarization system. Each classifier is used for judging whether each paragraph in the input text is essential or not.

Let us make some remarks on the connection of the text summarization with the text clustering. It is the process of segmenting a text group into subgroups based on

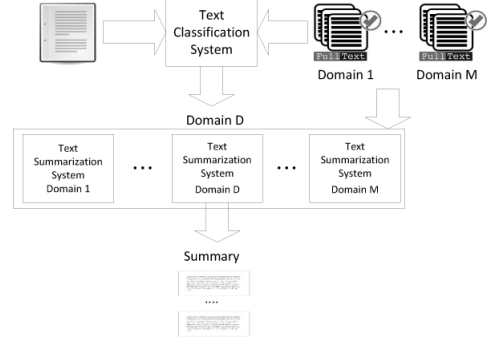


Figure 10. Text Summarization System connected with Text Clustering

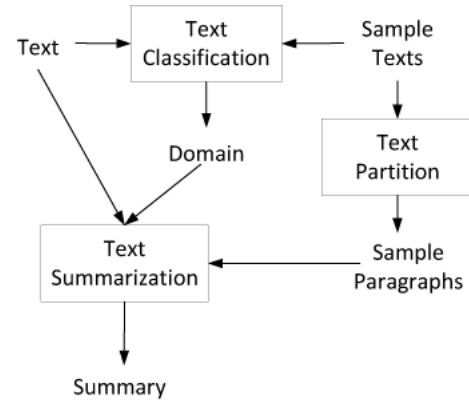


Figure 11. System Flow of Text Summarization connected with Text Clustering

their content-based similarities. The role of text clustering is to define the domains initially in the architecture of connecting the text summarization with the text clustering. In each domain, sample paragraphs are generated, and its corresponding classifier is trained with them. In the future research, we consider using the text summarization for clustering texts in the opposite direction.

REFERENCES

- [1] F. Sebastiani, "Machine Learning in Automated Text Categorization", pp1-47, ACM Computing Survey, Vol 34, No 1, 2002.
- [2] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text Classification with String Kernels", pp419-444, Journal of Machine Learning Research, Vol 2, No 2, 2002.
- [3] C. S. Leslie, E. Eskin, A. Cohen, J. Weston, and W. S. Noble, "Mismatch String Kernels for Discriminative Protein Classification", pp467-476, Bioinformatics, Vol 20, No 4, 2004.
- [4] R. J. Kate and R. J. Mooney, "Using String Kernels for Learning Semantic Parsers", pp913-920, Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, 2006.

- [5] T. Jo and D. Cho, "Index based Approach for Text Categorization", International Journal of Mathematics and Computers in Simulation, Vol 2, No 1, 2008.
- [6] T. Jo, "Device and Method for Categorizing Electronic Document Automatically", Patent Document, 10-2009-0041272, 10-1071495, 2011.
- [7] T. Jo, "Inverted Index based Modified Version of K-Means Algorithm for Text Clustering", pp67-76, Journal of Information Processing Systems, Vol 4, No 2, 2008.
- [8] T. Jo, "Representation of Texts into String Vectors for Text Categorization", pp110-127, Journal of Computing Science and Engineering, Vol 4, No 2, 2010.
- [9] T. Jo, "NTSO (Neural Text Self Organizer): A New Neural Network for Text Clustering", pp31-43, Journal of Network Technology, Vol 1, No 1, 2010.
- [10] T. Jo, "NTC (Neural Text Categorizer): Neural Network for Text Categorization", pp83-96, International Journal of Information Studies, Vol 2, No 2, 2010.
- [11] T. Jo, "Normalized Table Matching Algorithm as Approach to Text Categorization", pp839-849, Soft Computing, Vol 19, No 4, 2015.
- [12] T. Jo, "Graph based KNN for Optimizing Index of News Articles", 53-62, Journal of Multimedia Information System, 3, 2016.
- [13] T. Jo, "Graph based Version for Clustering Texts in Current Affair Domain", 171-176, The Proceedings of 15th International Conference on Data Science, 2019.